Defense Technical Information Center Compilation Part Notice

ADP014155

TITLE: A Data-Centric Infrastructure for Multidisciplinary Analysis Integration and Management

DISTRIBUTION: Approved for public release, distribution unlimited
Availability: Hard copy only.

This paper is part of the following report:

TITLE: Reduction of Military Vehicle Acquisition Time and Cost through Advanced Modelling and Virtual Simulation [La reduction des couts et des delais d'acquisition des vehicules militaires par la modelisation avancee et la simulation de produit virtuel]

To order the complete compilation report, use: ADA415759

The following component part numbers comprise the compilation report:
ADP014142 thru ADP014198

A Data-Centric Infrastructure for Multidisciplinary Analysis Integration and Management

Jean-Yves Trépanier François Guibault Benoît Ozell
CERCA (Centre de Recherche en Calcul Appliqué)
5160 Boul. Décarie, Suite 400
Montréal, Québec, Canada, H3X2H9

email: firstname.lastname@cerca.umontreal.ca

Abstract

This paper presents the overall architecture of the VADOR application framework. The purpose of the VADOR framework is to enable the seamless integration of commercial and in-house analysis applications in a heterogeneous, distributed computing environment, to allow the deployment of automatic design optimisation algorithms and to provide a comprehensive data-management infrastructure for design and analysis data. The emerging database will then provide the basis of a future knowledge-based engineering system and will allow IT links with other IT systems in the enterprise. A multi-tiered client-server architecture has been devised, which comprises a client GUI for interactive data definition and execution launching, separate data and execution servers, and autonomous remotely executable application wrappers.

1.0 Introduction

Computational-based design, including computational fluid dynamics (CFD) and computational structural dynamics (CSD), to name a few, are key technologies in the aerospace industry and are now part of the daily work of engineers. However, such a highly complex computational-based design environment, composed of a mix of commercial and "in house" programs tailored to the specific requirements of the design and analysis tasks, the application of multidisciplinary analysis and optimisation (MDO) practices faces a number of significant challenges, including integration, collaboration and data sharing and management. In this context, there is a need for a software infrastructure which will integrate heterogeneous applications, enable data sharing and collaboration, and insure proper data management.

In addition to these requirements for MDO applicability, there is currently a broad effort in organisations to leverage information technologies in order to enhance the access to information at all levels in the enterprise. Large companies are deploying ERP and PDM systems to better organise their information. At the technical engineering level, information is often residing in the engineers' experience and a system which could capture this information could be of a great benefit for the engineering departments as well as for the enterprise.

These two needs are at the heart of the VADOR (Virtual Airplane Design Optimisation framewoRk) project, and the specific objectives of the project are:

- To develop a state-of-the-art software framework capable of supporting an MDO paradigm in a collaborative design environment.
- To develop a comprehensive data-management infrastructure allowing to closely follow the design data used and shared by the design team and which will provide the basis of future knowledge-based engineering systems and IT links with other IT systems in the enterprise.

Paper presented at the RTO AVT Symposium on "Reduction of Military Vehicle Acquisition Time and Cost through Advanced Modelling and Virtual Simulation", held in Paris, France, 22-25 April 2002, and published in RTO-MP-089.

2.0 Framework Design and Architecture

The VADOR framework is an object-oriented infrastructure which manages files, groups of files, programs and processes and maintain a database which stores the links between these basic objects. The next section presents the main characteristics of the framework.

2.1 Data files

Objects called ***DataComponent***, or ***DC*** encapsulate the data files in the framework. An ***AtomicDC*** encapsulates one data file of a given type. The description of more complex groups of files is performed by allowing user-defined hierarchical composition of ***DataComponents***. The definition of ***Composite DataComponent***, or ***CDC***, contains a list of ***AtomicDC*** types or ***CompositeDC*** types. ***DC***'s and ***CDC***'s encapsulate the file references (URL) and a large set of attributes required for data management which are all stored in a relational database.

2.2 Programs and processes

VADOR provides a model for the encapsulation of the programs and processes used to create the data and provide mechanisms to logically link data and processes at an abstract level. A program, which can be any piece of software requiring some data files as input and producing some data files as output, is encapsulated in an object called ***AtomicStrategyComponent*** A process, which is an assembly of programs to be executed in a controlled sequence of operations, is encapsulated in a ***CompositeStrategyComponent***. Programs and processes are fundamentally defined in relation with the type of ***DC*** that they produce and the type of ***DC*** they require as inputs.

2.3 Distributed Architecture

The distributed architecture of the VADOR system is illustrated on Figure 1. The main elements of the system are: the Graphical User Interface (GUI), the Librarian Server, the Executive Server, the CPU Servers and the Database Management System (DBMS).

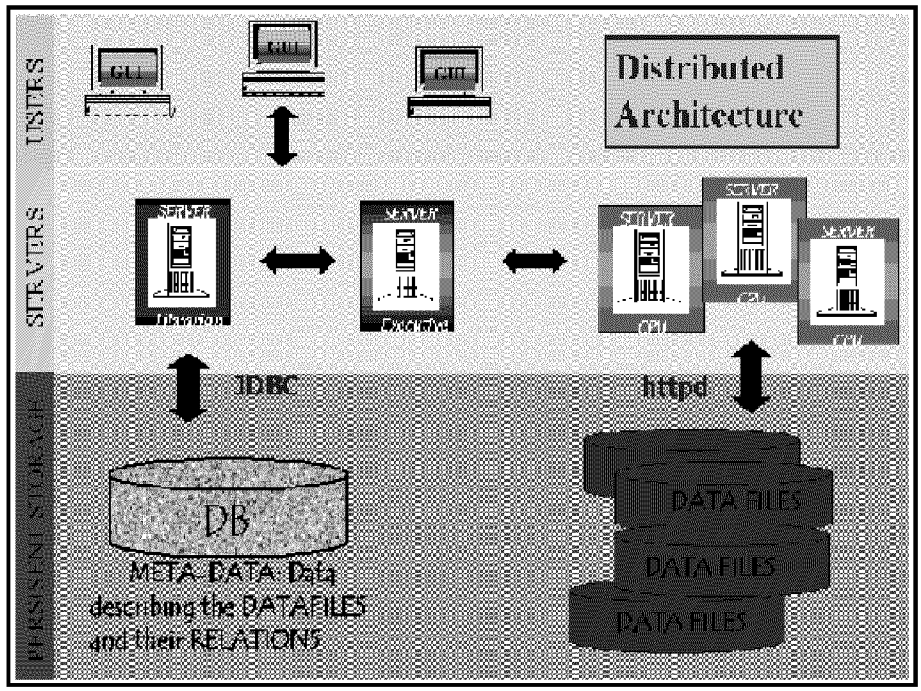


Figure 1 The distributed architecture of the VADOR system

2.4 Graphical User Interface

The VADOR GUI, shown in Figure 2, is a Java program running on the user's machine which provides an interface between engineers and the VADOR services. VADOR provides two main classes of services: the first class of services concerns the data and process definition and the tools registration. This is performed via four different tools called ***BUILDERS***. The second class of services include data management services, automatic execution of processes and data inspection tools. These two classes of services are described below.

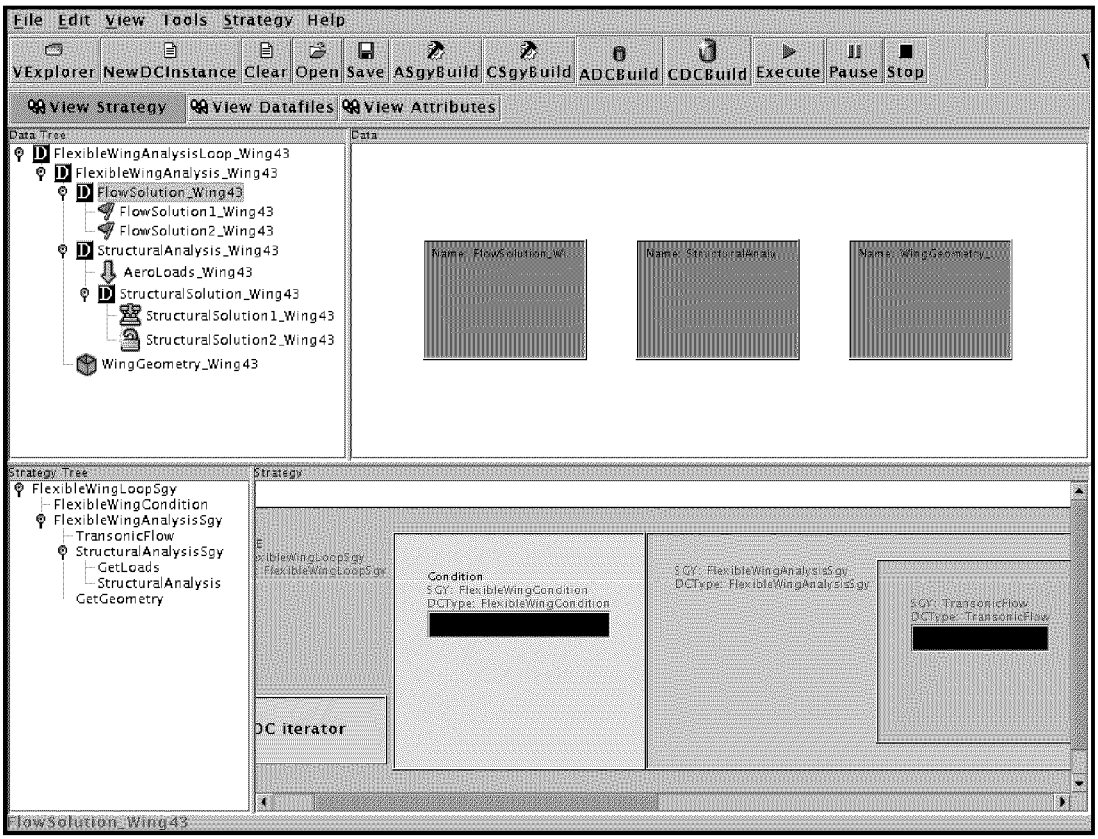


Figure 2 The VADOR Graphical User Interface

2.4.1 Builders tools for data and process definition

The GUI provides tools to define ***DataComponents*** type and to define ***StrategyComponents***. The tool used to define an ***AtomicDataComponent*** type, called the AtomicDCBuilder, is illustrated on figure 4. A different dialog box, called the CompositeDCBuilder, illustrated on figure 5, is used to construct a hierarchical composition of data into a tree structure. The tool used to encapsulate programs into ***AtomicStrategyComponents*** is called the AtomicStrategyBuilder and is shown on figure 3. A separate window, called the Composite Strategy Builder is used to describe processes including loops and if constructs. A view of the Composite StrategyBuilder is given on figure 6. On this figure, a strategy consisting of three sequential programs is modelled, each program having its own inputs and producing an element of a hierarchical ***DataComponent*** shown in a tree view on the left. The complete strategy is producing a ***Composite DC***.

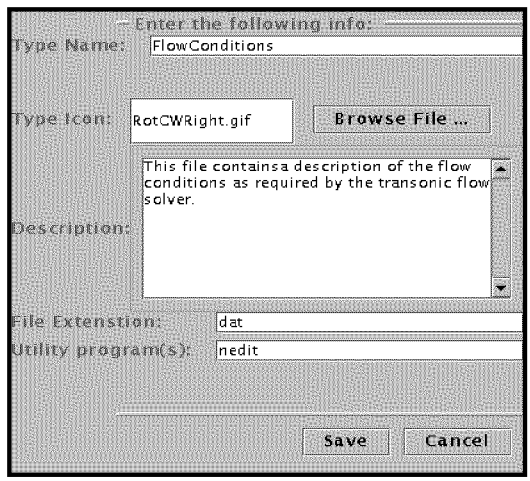


Figure 3 The Atomic Data Component Builder

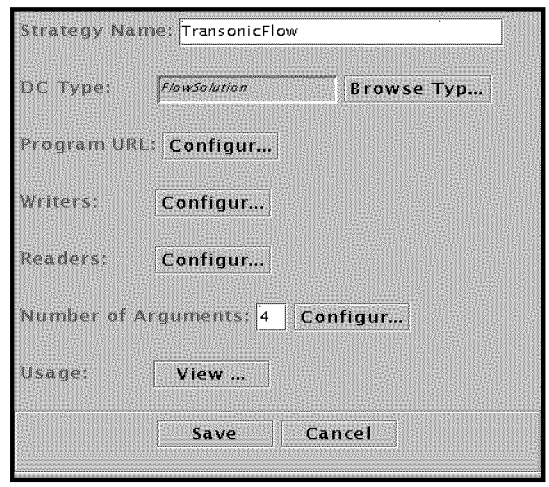


Figure 4 The Atomic Strategy Builder

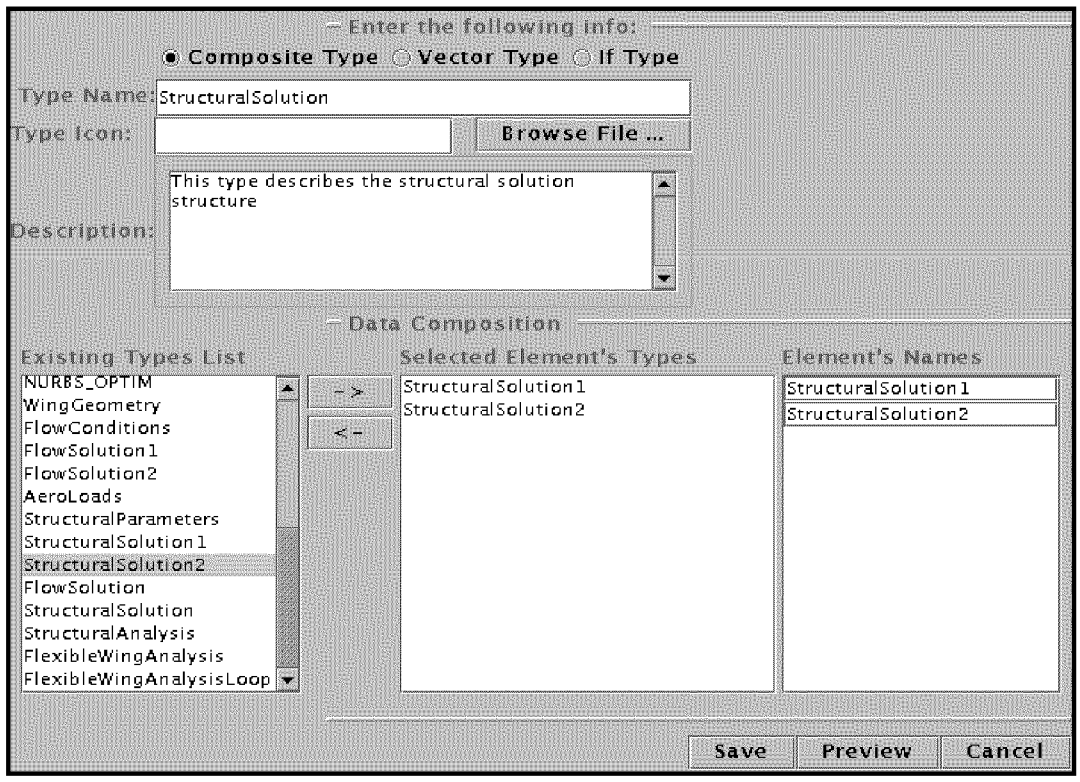


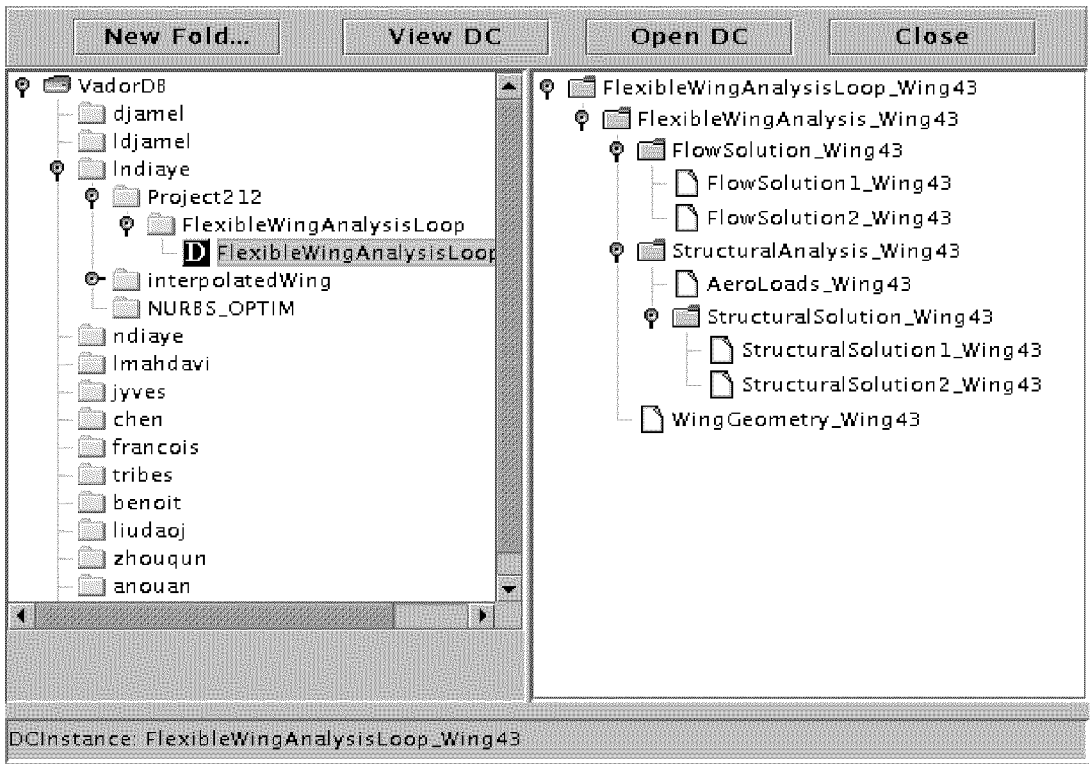
Figure 5 The Composite Data Component Builder Dialog Box

2.4.2 Data Management, Automatic Execution and Inspection Services

The data management services provided by VADOR includes a data classification layer and an automatic naming scheme for ***DataComponents*** and data files created under the system. A view of the VadorExplorer, the GUI window giving access to the classification layer, is shown on the figure below.

Using the information encapsulated in the ***DataComponent*** attributes, VADOR allows users to trace the upstream history of a given piece of data including the creator of the data, the programs used to create the data and the input data used. Users can also be informed on the downstream influence of a given piece of data by asking which data has been produced using a given piece of data as input. These services results in a comprehensive documentation about data dependencies and data influences.

Design and analysis work is performed in the system by first selecting the type of data to create in a user-defined list and then by selecting a strategy to create the data. Only the strategies which have been defined in the definition phase for the type of data to be created will be presented to the user. Once the data type and strategies are known, the system will generate a new instance of data, pointing to empty data files, and will ask the user to complete the definition of input data files to the programs. Again, files will be requested for various types, as described in the integration phase. After the selection of input files, the process is ready for execution and the user can select machines where the programs are to be executed and launch the execution. The control will then be passed to the executive server until the process execution terminates.



Referring to Figure 2, the upper half of the main window provides a graphical representation of a ***DataComponent***, where a tree representation is used to display on the left the composition of a hierarchic ***DataComponent***. The leaves of the tree represent the actual data files and their type can be easily identified by the corresponding icon. Individual rectangles shown in the right window identify individual ***DataComponent***, enabling selection and queries to be made on their contents. Utility programs, defined in the DCBuilder tool, can be invoked from the menu or the mouse in order to edit or visualise the contents of the encapsulated data files.

The lower half of the main window provides a graphical representation of the ***StrategyComponent***, where a tree representation is used to display the composition of a hierarchic ***StrategyComponent***. The complete ***StrategyComponent*** is graphically represented by recursively including ***StrategyComponent*** within each other.

2.5 Librarian Server

The Librarian Server is the central Component server in the VADOR system. The Librarian is providing services for the handling and archival of Components. The Librarian stores permanently the components in a relational database using the JDBC driver. Details about the organisation of the database are given below.

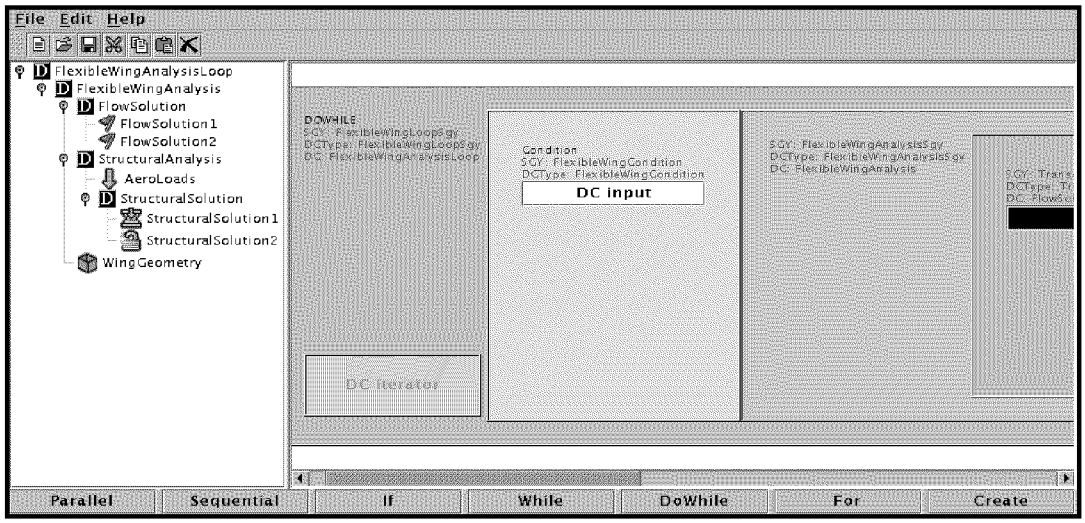


Figure 6 A view of the Composite Strategy Builder used to define processes

2.6 Executive Server

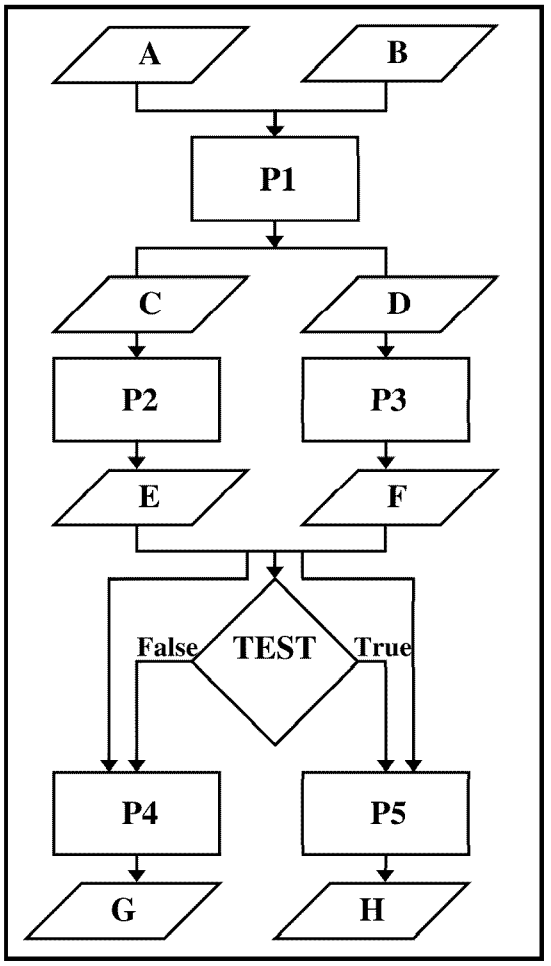
The Executive is a Java server program that manages the execution of ***StrategyComponents*** to create ***DataComponents*** on a distributed network of heterogeneous computers. It answers the needs of process automation in an heterogeneous distributed environment. The Executive is a multithreaded server capable of handling multiple tasks. The Executive interacts with the Librarian to retrieve ***DataComponents*** to be created and to update the database contents after execution. The Executive usually receives simple requests from the GUI to create ***DataComponents*** and it communicates with the Librarian to retrieve the ***DataComponent*** object. It traverse the Strategy Tree to generate the data creation sequence and it communicates with the CPU Servers on different hosts to run the analysis programs. During the execution process, it notifies the Librarian server after execution of each program in order to update the status of the components in the database.

2.7 CPU Servers

CPU servers, also called wrappers, are Java server programs waiting for requests on machines where analysis programs are to be run. The CPU servers wraps any scripts or executable programs on specific machines. The CPU Servers have the responsibility to run analysis programs with a list of I/O files and flags transmitted by the Executive Server. The result of the execution of the programs are the creation of the data files encapsulated in the ***DataComponents***. The CPU Servers also have the responsibility to get the input files required for the execution and put the output files to required locations after execution. File transfers are performed under the control of the CPU Servers using standard web servers for downloading and uploading files. The analysis programs are usually executable legacy programs to be executed on a specific machine, or a few machines, on the network. Note that the execution time required to run these programs can vary from a few milliseconds to many days, depending on the specific engineering analysis to perform.

3.0 Process example

A process in the VADOR framework is a controlled sequence of program execution which is used to produce complex data. Processes are encapsulated in Composite ***StrategyComponents*** objects which are defined by the users using standard elements of structured procedural programming languages: sequential blocks, parallel blocks, if constructs and controlled loops. The figure shows a typical process which will be used to illustrate the definition of a Composite ***StrategyComponent*** used to create a Composite ***DataComponent.*** In the figure, parallelepipeds indicate files and rectangles indicate programs.



First, we need to define the data produced by the process. This is done by defining the Atomic DC types A, B, C, D, E, F, G, H. We also need to define a composite data type C-PLUS-D to encapsulate the output of the program P1. Considering the fact that the A and B types are not part of the process but are rather inputs to the process, we need to define a composite DC type C-TO-H containing one C-PLUS-D, and one E,F,G and H respectively to encapsulate the output of the complete process. The composite DC type C-TO-H will encapsulate 6 data files.

The process is modelled as a sequential process composed of three blocks. In the first block of the sequence, the program P1 produces a C-PLUS-D type DC using an A type DC and a B type DC as input. In the second block of the sequence, the programs P2 and P3 are executed in parallel, the P2 program producing an E type DC using a C type DC as input and the program P3 produces an F type DC using a D type DC as input. The third and last block of the sequence contains an if construct with its associated True and False branches. The test uses input data contained in either or both the E and F DC types to branch. In the FALSE branch, the P4 program produces a G type DC using a E and a F type DC as input while in the TRUE branch, the P5 program produces a H type DC using the same input. It is clear that at the end of the process, either the G or the H type DC will be pointing to a non-existing data file.

Note that there is clear separation in the system between the sequence of execution and the dependencies. For example, the system allows one to describe a process which will use as input a file which will be produced later in the process. This provides flexibility in the description of complex process. However, at execution time, the process may not be able to execute and appropriate messages will be generated.

4.0 Relational Database

VADOR make use of a standard Database Management System (DBMS) to store and access information describing the various Components. The DBMS currently used by the framework is the MySQL DBMS, a publicly available system which stores information using a relational model using the standard SQL language for database queries. A view of the various tables and their relations is

reproduced using the UML representation on figure 7. It is important to emphasise that the present data model separates the engineering data usually contained in data files, from descriptive information. Only the descriptive information, or metadata, is stored in the relational database. The users' data files (potentially large files) will usually reside where they have been created by the application programs.

Devising a schema for a relational data model is not a simple task and the tables contents and their relations as currently described in figure 4 are the results of a few iterations and will probably require adjustments in the future. The final schema, or data model, will be one of the main results of the VADOR project since it defines the core of the system on which everything else is built. It also defines the information that the system will be able to provide to other IT systems through appropriate APIs.

4.1 Data and Process Definition Tables

Referring to figure 4, the first table, called the DCDefinition Table stores the definition of Atomic and Composite DataComponents Types. When the DC type is Composite, the elements of the composition are listed in the Table DCElements. The DCDefinition table identifies each type of data in the VADOR system by a unique typeID, known as the Primary Key (PK) of the table and associates with this typeID a typeName, a typeIcon, an IsAtomic field and a Description field. The typeName is given by the user and is required to be unique. The typeIcon refers to an icon file managed by the system which will be used by the GUI to provide visual recognition of the ***DataComponents*** based on their type, as shown in figure 2. The IsAtomic field allows to distinguish between Atomic and Composite DC types. The Description field is a string given by the user at the moment of the type definition which will be later accessible through the GUI for information on types. Composite DC are defined in the table DCElements. Every element appearing in a composite DC type has a unique dcElementID, has a dcElementName, has a ChildID, has a counter and has a typeID. The typeName should be chosen carefully to match engineering practices and an Icon should be designed for visual support.

Programs and processes are defined in the table named StrategyDefinition. A relation is present between the StrategyDefinition Table and the DcDefinition Table, indicating explicitly the DC Type that the strategy can create. A Composite StrategyComponent is described by its elements stored in the StrategyElements Table.

The above described four table are the core of the data and process definition in the VADOR framework. Note that users will not in general need to deal with the database tables but only with the GUI presented previously.

4.2 MetaData Tables

The central table for metadata storage and management is the DcInst table which stores instances of ***DataComponents***. DC Instances encapsulate single data file or groups of data files and attach to these files metadata information. The table has two fundamental links with the Definition tables. First, the DcInst table has a relation with the DcDefinition table in order to uniquely define the type of data described by the instance. Second, the DcInst table stores the strategy used to create the data through a relation with the StrategyDefinition table. The DcInst table contains general descriptive information about the instance required for data management. For a composite DC type, the elements instances of the composition are listed in the DcElementsInst table.

The DcInst table also has numerous relations with various tables in order to provide additional information and control on the data. Relations with the DcAccess table enable a flexible control on access permission to the data on a user base. Relations with the ExternalInputDcInst table allow to uniquely identify the data used as input to processes and thus provide comprehensive forward and backward dependencies capabilities. The ExecutionInfo table will store information about the execution statistics and will enable future deployment of load balancing capabilities. Relations with

the DcFolder enables the efficient classification of the data based on user's defined projects through the VADOR Explorer.

5.0 Conclusion and prospective

The object-oriented methodology has been used in the development of a data-centric framework. The implementation is done using the JAVA computer language. Collaboration and data-sharing are enabled through the usage of the concept of Components and through data standardisation. Two kinds of components have been defined: DataComponents and StrategyComponents. DataComponents encapsulate the design-and-analysis data while StrategyComponents encapsulate the design-and-analysis workflow. In order to provide data management capabilities, the components have a set of attributes, including the owner, access permission, history, comments, status and more. In order to promote standardisation, user defined data types are introduced, which are then used for validation and documentation. DataComponents and StrategyComponents are hierarchically arranged, i.e. they may contain other components, giving the capability to describe the most complex problems. The leaves of this composition are called the atomic components. The atomic DataComponents encapsulate exactly one data file, while the atomic StrategyComponents encapsulate one program or script.

The VADOR system results from the assembly of distributed servers, allowing scalability in a multiple users environment. The Librarian server provides access and storage services for DataComponents and StrategyComponents. The Executive server controls the execution of sequences of analysis programs. The CPU servers, or wrappers, run analysis programs. File transfers are performed via standard Web file servers. A relational database stores data references and attributes as well as data creation strategies. VADOR uses standard SQL databases for portability.

The current version of the VADOR framework is a flexible and configurable software, adaptable to the needs of every engineer. It is capable of representing information, including data and methods, uniformly from the enterprise work flow charts to the detailed engineering tasks. It allows process execution automation on a distributed network and enables sharing of data and methods while providing critical information to all users on the location and owner of data and methods, the methods used to produce the data and the status of data and tasks. The integration of data and processes into the framework enforces documentation of data formats and methods, and as a result promotes standardisation.

The VADOR team is currently implementing various additional services in the framework, including a load monitor and balancing facility, a data register where users can indicate their interests on specific events related to data or strategies, and the generic implementation of optimisation strategies. Future directions include a comprehensive support of generic design strategies, including formal decomposition methodologies for MDO problem formulation and optimisation. Efforts will also be deployed to define and use standard data descriptions for seamless data exchange between heterogeneous applications.

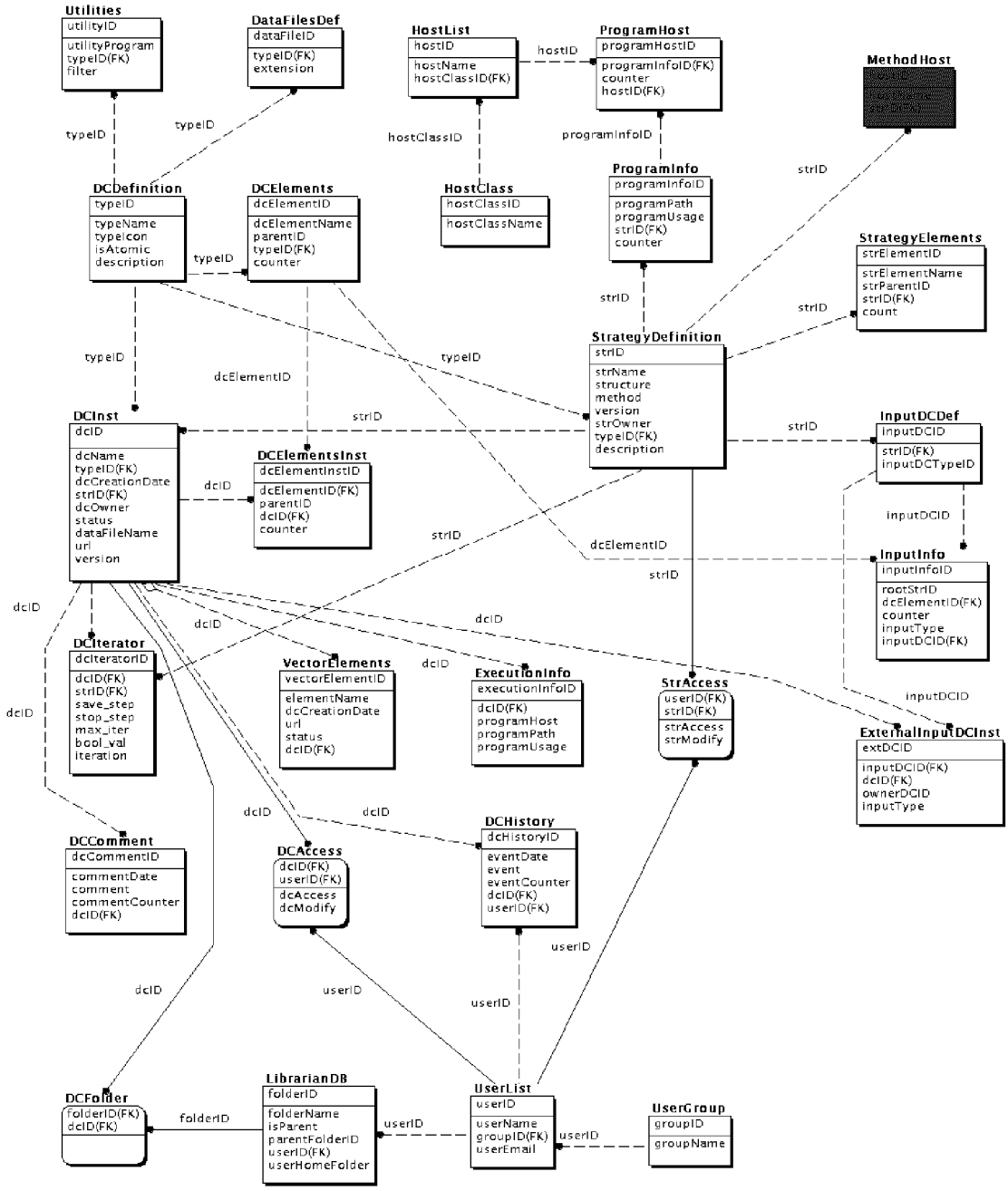


Figure 7 Relational Database Tables and relations

Paper #10
Discussor's name M. Stevenson
Author J. Y. Trepanier

Q: 1) How is geometry handled in the system?
2) How do you handle computational requirements vs. database complexity?

A: 1) The system is managing files. Geometry files, such as CATIA files, can be referenced by the system as external input files. Then, specialized translation programs are used to extract data from CAD files and produce the files required for analysis work. In some cases, the process of preparation of input files for analysis from the CAD files is manual.
2) The system will store computational requirements for each task performed under the control of the system based on information gathered during execution. This information will constitute a rich database enabling the correlation of computational requirements with data and process attributes. This will then be used to estimate CPU requirements for a given task and to feed a load balancing facility.